Amendments to the Specification

Please replace the paragraph on Page 1, lines 4 - 10 with the following marked-up replacement paragraph:

-- The present invention is related to U. S. Patent _____, titled "Machine-Oriented Extensible Document Representation and Interchange Notation" (serial number ~~09/_____~~, ~~referred~~ 09/652,056), referred to herein as the "first related invention", and U. S. Patent _____, titled "High-Performance Extensible Document Transformation" (serial number ~~09/_____~~, ~~referred~~ 09/653,080), referred to herein as the "second related invention", both of which were filed concurrently herewith. These related inventions are commonly assigned to International Business Machines Corporation (IBM), and are hereby incorporated herein by reference. --

Please replace the paragraph on Page 2, lines 10 - 20 with the following marked-up replacement paragraph:

-- Business and consumer use of distributed computing, also commonly referred to as network computing, has gained tremendous popularity in recent years. In this computing model, the data and/or programs to be used to perform a particular computing task typically reside on (i.e. are "distributed" among) more than one computer, where these multiple computers are connected by a network of some type. The Internet, and the part of the Internet known as the World Wide ~~Web (hereinafter, "Web", are~~ Web (hereinafter, "Web"), are well-known examples of this type of environment wherein the multiple computers are connected using a public network. Other types of network environments in which distributed computing may be used include intranets, which are typically private networks accessible to a restricted set of users (such as

employees of a corporation), and extranets (e.g., a corporate network which is accessible to other users than just the employees of the company which owns and/or manages the network, such as the company's business partners). --

Please replace the paragraph that begins on Page 3, line 14 and carries over to Page 4, line 6 with the following marked-up replacement paragraph:

-- The syntax of XML is extensible and flexible, and allows document developers to create tags to convey an explicit nested tree document structure (where the structure is determined from the relationship among the tags in a particular document). Furthermore, document developers can define their own tags which may have application-specific semantics. Because of this extensibility, XML documents may be used to specify many different types of information, for use in a virtually unlimited number of contexts. It is this extensibility and flexibility which is, in large part, responsible for the popularity of XML. (A number of XML derivative notations have been defined, and continue to be defined, for particular purposes. "VoiceXML" is an example of one such derivative. References herein to "XML" are intended to include XML derivatives and semantically similar notations such as derivatives of the Standard Generalized Markup Language, or "SGML", from which XML was derived. Refer to ISO 8879, "Standard Generalized Markup Language (SGML)", (1986) for more information on SGML. Refer to "Extensible Markup Language (XML), W3C Recommendation 10-February-1998" which is available from the World Wide Web Consortium, or "W3C", for ~~on the World Wide Web at http://www.w3.org/TR/1998/REC-xml-19980210, for~~ more information on XML.) --

Please replace the paragraph on Page 6, lines 1 - 14 with the following marked-up replacement paragraph:

-- Furthermore, prior art techniques for storing XML documents and processing those documents require a considerable amount of processing and storage overhead. Typically, an XML document is parsed and stored internally as a Document Object Model (DOM) tree representation by an XML parser. DOM trees are physically stored in a tree representation, using objects to represent the nodes in the tree, the attributes of the nodes, the values of the nodes, etc. Operations are then performed (e.g. by content renderers or style sheet processors) then operate upon this tree representation. For example, deleting elements from a document may be accomplished by pruning subtrees from the DOM tree; renaming elements within a document may be accomplished by traversing the DOM tree to find the occurrences of the element name, and substituting the new name into the appropriate nodes of the DOM tree. (DOM is published as a Recommendation of the W3C, titled ~~World Wide Web Consortium ("W3C"), titled~~ "Document Object Model (DOM) Level 1 Specification, Version 1.0" (1998) ~~and available on the Web at http://www.w3.org/TR/REC-DOM-Level-1. "DOM" is a trademark of Massachusetts Institute of Technology.~~) --

Please replace the paragraph on Page 21, lines 12 - 17 with the following marked-up replacement paragraph:

-- With more and more application programs being written to operate upon XML documents, the improvements yielded by this array-based storage representation will have a significant impact. (The array-based approach disclosed herein may also be used advantageously

for representing documents that have been encoded in other structured document notations, and thus references herein to using the arrays of the present invention for XML and mXML ~~documents is intended~~ documents are intended for purposes of illustration and not of limitation.) --

Please replace the paragraph on Page 23, lines 1 - 12 with the following marked-up replacement paragraph:

-- Fig. 4A illustrates a simple structured document 400 which is represented in the existing XML notation. This document contains 6 elements which are organized in a 3-level hierarchy. The Node having element name "root_element" 402 is the root node, being at the highest level of the hierarchy. This node has 2 child nodes, having element names "level_one_element1" 410 and "level_one_element2" 420. Node "level_one_element1" 410 also has 2 child nodes, which are the nodes having element names "level_two_element11" 412 and "level_two_element12" 414, and ~~node "level_two_element2" 420~~ node "level_one_element2" 420 has a single child node having element name "level_two_element21" 422. A tree structure 430 representing document 400 is shown in Fig. 4B, where the tags for the 6 elements are depicted inside rectangular shapes representing nodes of the tree and the data content corresponding to each node is shown inside an ellipse. This interpretation of an XML document 400 and its corresponding tree structure 430 are well known in the art. --

Please replace the paragraph that begins on Page 38, line 18 and carries over to Page 39, line 3 with the following marked-up replacement paragraph:

-- The test in Block 815 asks whether ~~there are more~~ all nodes ~~to be processed~~ in the source mXML have been processed. This test has a negative result when N_Count is less than or equal to the node count from the source mXML. In this case, processing continues at Block 820; otherwise, when the test has a positive result, the parsing process is complete, with the arrays being fully constructed, and the processing of Fig. 8 ends (Block 890). --

Please replace the paragraph that begins on Page 38, line 18 and carries over to Page 39, line 3 with the following marked-up replacement paragraph:

-- Block 835 scans to find this mXML element's children list in the mXML source. The children list begins after the semi-colon which ends the element's name, and continues up to the next semi-colon delimiter. The children list comprises a comma-separated list of integer values which identify the relative positions of the element's child elements. Block 840 asks whether the ~~there were any~~ children list was empty. This test has a positive result when 2 semi-colon delimiters appear in sequence, with no list of child elements. If there were no children, control transfers to Block 860; otherwise, Blocks 845 through 855 process the children list. --

Please replace the paragraph that begins on Page 44, line 6 and carries over to Page 45, line 1 with the following marked-up replacement paragraph:

-- Fig. 10A depicts a preferred embodiment of the logic which may be used to retrieve information from the arrays of the present invention, and to manipulate that information. This type of operation may be performed by a style sheet processor, by an executing application program, and so forth. As examples of retrieval operations that may be performed directly on an

array-based representation of a structured document, it may be necessary to determine an element's children or perhaps its parent. Or, the element's value or attributes may be needed. A typical process begins by searching through the element name array (Block 1000) to locate the element having a particular name. This comprises using the starting and offset values, along with the mXML buffer pointer, to find the text of the element name, and comparing that text to the particular name of interest. (Alternatively, the node index may be known in some cases, in which this search is not required.) Block 1005 then gets the node index of the element array entry with the matching name. (In some cases, multiple array entries may have the matching name, and the ~~indexes~~ index of each such entry is then obtained, for example by building a list of indexes, in Block 1005.) Block 1010 gathers the information about this element, such as its value from the element value array, its parent, its children, etc. This information is retrieved easily and efficiently using the located node index to index into the other arrays. An API may optionally be invoked, as shown at Block 1015, to manipulate this element's information. --